International Journal on Advanced Science Engineering Information Technology

Vol.13 (2023) No. 3
ISSN: 2088-5334

Position Data Estimation System Based on Recognized Field Landmark Using Deep Neural Network for ERSOW Soccer Robot

Iwan Kurnianto Wibowo [a,*], Mochamad Mobed Bachtiar [a], Erna Alfi Nurrohmah [a], Vega Kurnia Garindra Wardhana [a]

[a] *Department of Informatics and Computer Engineering, Politeknik Elektronika Negeri Surabaya, Surabaya, 60111, Indonesia*
Corresponding author: [*]eone@pens.ac.id

***Abstract*—One of the problems faced by soccer robots is how to find out the position of the robot itself and other robots on the field. A simple way to find out the robot's position is to use the odometry method. However, odometry is weak in accumulating position errors that reduce the accuracy of the moving robot's absolute position estimation and orientation. This paper presents a robot position data estimation system that is to be implemented on the ERSOW wheeled soccer robot. The robot can determine its position based on a unique landmark: an L-shaped line on the soccer robot field. We use a deep neural network method to recognize landmark L-shaped. Vision systems and deep learning inferences run on the Robot Operating System platform. After obtaining the distance of the robot to the L-shaped landmark, the robot's orientation and position relative to the field can be accurately determined based on the omnidirectional camera's perception. The results of the position estimation system in this study can be used to reduce position errors resulting from the odometry method. Based on the landmark L-shape recognition test results conducted on 641 datasets, the validation accuracy value is 0.806. The results of testing the robot position generated by vision obtained the largest error *x* about 2.32 cm and *y* about 1.99 cm.**

***Keywords*— ERSOW robot; estimation; landmark L-shape.**

Manuscript received 25 Oct. 2022; revised 13 Jan. 2023; accepted 18 May 2023. Date of publication 30 Jun. 2023.

I. INTRODUCTION

Robot soccer is one type of robot that has the task of playing football like humans [1]. There are two types of soccer robots, namely humanoid soccer and wheeled soccer. Soccer robots have basic abilities to find the ball, dribble, kick the ball, find the goal, recognize opponents and friends, avoid collisions, and coordinate with each other. The soccer robot also requires self-localization skills to determine its field position [2]. A reliable robot position and orientation estimation system is needed to support these basic abilities. Odometry is a technique that uses the robot's relative movement and the measurement of the angular rotational speed of its wheels to estimate the robot's position. Odometry's primary drawback is the accumulation of errors, which decreases the precision of estimates of the moving robot's absolute position and orientation [2].

The ERSOW robot is one of those wheeled soccer robots that has all the major individual skills. One of the most important skills is knowing the robot's orientation and position on the field [3]. ERSOW also uses the odometry method to assist the position estimation system. The orientation of the robot can be easily established using the IMU sensor or compass [4]. However, in the implementation in the field, there are many errors of angular drift in the IMU and also the bad influence of the magnetic field of the robot actuator itself. So that the robot often experiences orientation disturbances and the wrong position. The orientation of the robot affects the positioning of the robot [5]. For a long time, the ERSOW robot determines orientation using the IMU sensor. The ERSOW robot often has orientation problems because the IMU sensor has an accumulated drifting effect. So, it requires periodic orientation and position calibration [6]. The odometer calibration is done by detecting a line in the field using a line sensor placed under the robot. In research by Romadon [7] and Alatise and Hancke [8], Kalman Filter is used to estimate a robot's orientation and position. When the robot moves slowly with a fixed orientation, it will produce a good estimate of the position and orientation. However, when the robot has to move quickly above 110 cm/s with a rapid change in the robot's angle, it will result in a large orientation

and position estimation error. The ERSOW robot runs at speeds above 100cm/s, so this is considered insufficient.

According to Luo et al. [9], to feed the ball to a friendly robot, a localization system is needed to determine the position and orientation of the robot itself and the friend robot in the field. Position localization is based on 3D positioning from the Kinect camera by processing the depth point cloud value[10]. The method used to identify other robots is Convolutional Neural Networks which run on the GPU. Point cloud depth is combined with 2D positioning to get real-time 3D positioning.

The study of Karkoub et al. [11] combines a conventional camera with a hyperbolic mirror. Thus, the vision becomes wide and looks omnidirectional. The final image quality is highly dependent on the shape of the mirror and the camera or mirror settings. To provide a 360° field of vision, mirror play a very important part in this vision system in the MSL RoboCup [12]. Omnidirectional robot often uses a catadioptric vision system. The camera module set consists of an upright camera facing a convex mirror. The mirror used, in general, is parabolic or hyperbolic. Engel et al. [13] study explains that a camera's RGB image can be regressed using a convolutional neural network. The deep learning approach promises better results in recognizing landmark objects. Landmarks can be used for localization, making it a useful map. One of the deep learning methods, DNN, can be trained quickly and give good test results [14], [15]. Research by Dwijayanto et al. [16] uses YOLO to detect landmarks on the soccer field in the Indonesian robot contest. However, not yet able to estimate the position. The fastest version of YOLOv3 is YOLOv3-tiny. The basic difference from YOLOv3 is in the total layer used. To detect the system, YOLOv3-tiny uses 24 layers. OpenCV is a framework developed for real-time computer vision programming [17]. In this regard, Google developed an end-to-end open-source machine learning platform using OpenCV called TensorFlow[18]. TensorFlow provides a deep neural network solution and can be considered as an advanced framework for object detection [19], [20], including robot application detection [21]. The training model greatly affects the detection test results. Large-scale datasets are needed to get better training model results in learning visual features from images [22].

In this paper, we used L-shape line landmarks to assist the orientation estimation process in the ERSOW soccer robot. The L-shape line landmark is unique compared to other line shapes on the field, and this can reduce the potential for errors in landmark detection. The L-shape landmark is the outermost goal line located in front of the goal. Robots will often traverse the area, especially defender robots and goalkeepers. There are two L-shape landmarks in front of the goal. Camera with omnidirectional mirrors is used to take pictures of the field. The camera mounted on top of the ERSOW robot can see landmark lines up to 5 meters away. The recognition of L-shape landmarks were carried out using the Deep Neural Network.

II. Materials and Method

This research was built in several stages, as shown in Fig. 1. The vision system and artificial intelligence on the ERSOW robot are run using Robot Operating System (ROS). ROS manages all communication between the vision system, artificial intelligence, low-level control, and robot coordination [23]. The camera capture process and field landmark recognition are synchronized through ROS nodes so that the robot is able to respond to changes in the field in real-time [24]–[26].

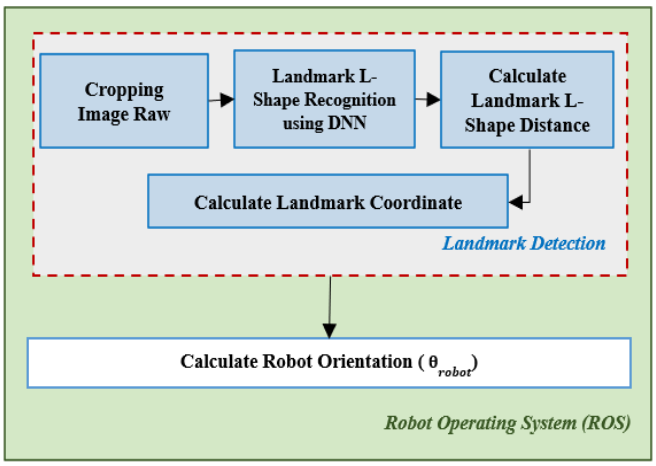

Fig. 1 ERSOW Robot Vision System

ERSOW vision system runs on a node in the ROS with a processing speed of 70 fps. Node vision manages several processes of taking pictures, pre-processing, landmark recognition using DNN, and calculating the distance of the landmark to the robot.

A. Image Pre-processing

At the publisher node, the camera captures the image with a resolution of 1280x720 pixels. The original image from the camera has an RGB color space.

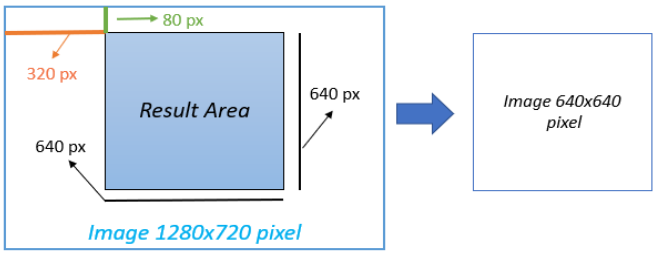

Fig. 2 Image cropping from 1280x720 to 640x640 pixel

As shown in Fig. 2, 1280x720 pixels image is cropped into an 640x640 pixels image. Remove parts of the image that are unnecessary, such as images outside the soccer field that can become noise.

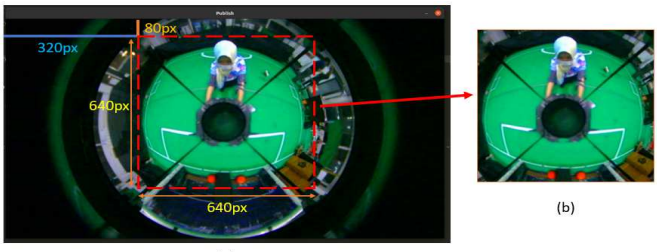

Fig. 3 Image cropping results. (a).Original image captured by camera. (b).Cropped image

Image cropping is extracting certain segments of an image. The method is quite easy, namely by slicing the image array. According to Fig. 3 (a), the image is extracted at the center of the omnidirectional mirror image. The image cropping process removes the image as many as 330 pixels for the X axis and 80 pixels for Y axis. The image is cropped at 0 pixels to 330 pixels for X axis. For Y axis, the image is cropped from 0 to 80 pixels. The cropped result is an image with a resolution of 640x640 pixels as shown in Fig. 3 (b). Thus, the remaining pieces are no longer needed. The publisher node sends the image to the vision localization node using image transport for further processing.

B. Data Preparation

The dataset was collected using image data in the PENS soccer robot field. The total number of landmark L-shape datasets is 641 images each, as shown in Fig. 4. Image resolution after going through the pre-processing stage is 640x640 pixels. The dataset is split into 620 data for training data and 21 data for testing data. The collected datasets need to be labeled to create a high-quality dataset for model training [27]. Image labeling focuses on identifying and marking details where the L-shape line landmarks are in an image. The images were reviewed manually and labeled in the form of bounding boxes in the area of the L-shape line landmarks.

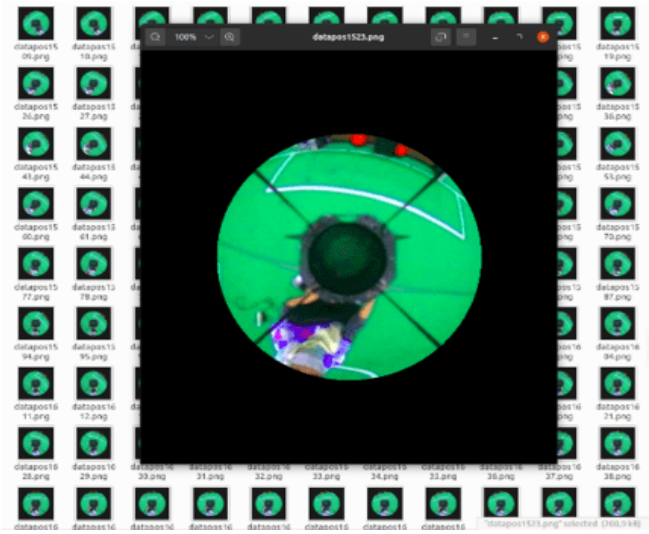

Fig. 4 Landmark L-shape dataset

C. Landmark Recognition Using Deep Neural Network

After the pre-processing stage is complete, the vision system will recognize L-shape landmarks in real time using the Deep Neural Network. Based on the calculation of the distance between the robot and the L-shape landmark, the coordinates of the L-shape landmark and the robot's orientation to the field can be determined.

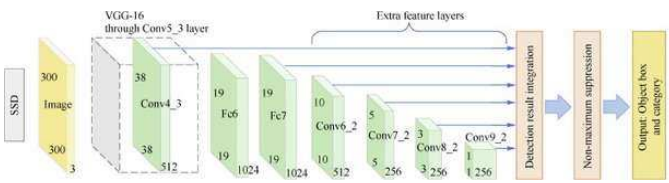

Fig. 5 Model of SSD Algorithm

SSD is an object detection algorithm created by Google with VGG16 (OxfordNet) as architecture. Fig. 5 shows the model of SSD Algorithm. SSD predicts objects using multiple prediction boundary boxes or often referred to as Multibox [28]–[31]. SSD speeds up the detection process by reducing the use of Region Proposal Network (RPN) 22. RPN is the backbone used to detect objects. SSD uses multi-scale features and default boxes to improve accuracy [11].

ERSOW robot applies machine learning algorithms to build models and predictions. The training involves using the deep learning framework that is SSD-MobileNet V2 with a tensor model. In the model training phase, entering large amounts of data into the DNN requires powerful computations. Here the role of the GPU is needed to shorten the training time. In the DNN training for ERSOW robot landmark recognition, three stages must be passed: loading the landmark dataset, training landmarks using DNN, and saving the training model. 2 data sets of L-shape line landmarks in front of the goal totaling 641 images each. Continued with the dataset training process using DNN with TensorFlow framework. Training a model requires specifying many parameters. But not all parameters will be used for inference. Tensorflow is able to identify the parameters needed for inference in the trained model. The output of the training results in the form of .pb and .config file models, which will then be run at the inference phase.

After the training phase is complete and produces the best DNN model, the inference phase is next. Pre-trained models loaded using the DNN module of the Single Shot Detector (SSD-MobileNet V2) deep learning framework. Fig. 6 shows Landmark Detection Process in ERSOW robot inference. Camera-captured images are used as input for the inference system where the image display is a reflection of the field via an omnidirectional mirror.

Fig. 6 Landmark Detection Process

The vision-localization ROS node runs a landmark detection program. The program calculates the landmark image's blob value and displays the predicted result on the frame. The output of object detection is the object's pixel coordinates. An object detection method works to detect an object in an image. Next, the location of the object in the image frame can be detected. The bounding box determines the location of the object. Models can make live data-driven

predictions to produce actionable results. There is a challenge in minimizing latency issues during the inference process to make decisions in real-time. The *landmarkRecognition* function is executed based on a *ROS::rate* loop rate(frequency) of 5 Hz.

```
LandmarkRecognition()
1: ROS::rate loop_rate = 5
2: While ROS == run
3:   DNN blobFromImage (readyframe, 1.0, Size(300,300))
4:   Net set input blobimg
5:   recognition = net forward ("detection_out")
6:   Mat(recognition.size[2], recognition.size[3])
7:   conf_threshold = 0.5
8:   for i=0 to recognitionMat.rows do
9:     detect_confidence = recognitionMat at (i,2)
10:    if detect_confident > conf_threshold then
11:      X1 = recognitionMat at (i,3) * readyframe.cols
12:      Y1 = recognitionMat at (i,4) * readyframe.rows
13:      Width2 = recognitionMat at (i,5) * readyframe.cols
14:      Height2= recognitionMat at (i,6) * readyframe.rows
15:      Rect rec(X1, Y1,(widthe2-X1), (Height2-Y1))
16:    end if
17:  end for
18:  Do show Window readyframe
19:  waitkey:1 millisecond
20: end while
```

DNN::blobFromImage will detect existing blobs in a *'readyframe'* image with a scale factor = 1.0, and a spatial size = {300x300} pixels. The spatial size is set equal to the spatial size of the completed training phase.

Set input "*blobimg*" before passing to *'Recognize'* image. The blob's threshold is set to 0.5; anything below that value will be disregarded. The program performs a scan *along recognitionMat.rows*This is repeated to determine the detected object's pixel coordinates for x, y, width, and height. At the *x* and *y* coordinates that have been found, a bounding box can be made based on the width and height of the recognized object. The recognized object is displayed on the *readyframe* with a delay in the form of a *waitkey* of 1 ms. The outcome of L-shaped landmark recognition is seen in Fig. 7.

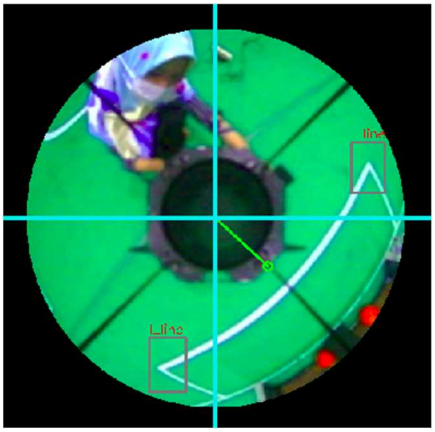

Fig. 7 Result of L-shaped Landmark Recognition

D. Landmark coordinate

The process to find the coordinates of the L-shape landmark begins with calculating the distance of the robot to the L-shape landmark. The distance between the landmarks to the robot can be calculated easily using mathematical equations. The values of *X1*, *Y1*, *Width2*, and *Height2* are the inputs of the Rect function. Next, it would be pushed back to rect_det. The *x*, *y*, *width*, and *height* coordinates of a rectangle are handled by the Rect function. The formula *center x + center w/2* may be used to determine the *x-coordinate* value for the *Xpixel* variable. While the *Ypixel* variable's y-coordinate value is determined using the formula *center y + center h/2*.

```
CalculateDistance()
1:  While ROS == run
2:  for i=0 to detectionMat.rows do
7:    X1 = recognitionMat at (i,3) * readyframe.cols
8:    Y1 = recognitionMat at (i,4) * readyframe.rows
9:    Width2 = recognitionMat at (i,5) * readyframe.cols
10:   Height2= recognitionMat at (i,6) * readyframe.rows
11:   rect_det.push_back(Rect(X1,Y1,Width2,Height2));
12: end for
13: sort (rect_det.begin(), rect_det.end(), compRect)
14: center_x = X1
15: center_y = Y1
16: center_h = Heigh2
17: center_w = Width2
18: Xpixel = center_x + center_w/2
29: Ypixel = center_y + center_h/2
20: Do calculate PixelDistance
21: Do Calculate RealDistance
22: end while
```

The distance between the robot and the landmark point can be found using the Euclidean equation as shown in the Fig. **8**.

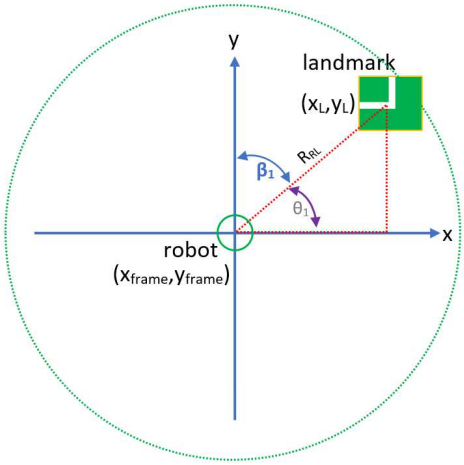

Fig. 8 View of landmark from camera with omnidirectional mirror perception

If the system can recognize the landmark and pass the validation, the landmark coordinates (X_L, Y_L) will be displayed, and a visual line will be drawn to the center point of the robot camera (x_{frame}, y_{frame}).

$$\boldsymbol{R_{RL}} = \sqrt{\left(\boldsymbol{X_{TL}} - \boldsymbol{x_{frame}}\right)^2 + \left(\boldsymbol{Y_{TL}} - \boldsymbol{y_{frame}}\right)^2} \quad (1)$$

Where R_{RL} is distance of landmark L-shape to camera center in pixel, (X_{TL}, Y_{TL}) is coordinate of landmark L-shape. (x_{frame}, y_{frame}) is coordinate of center camera on image frame. The value of the β_1 angle is obtained using equation (2).

$$\boldsymbol{\beta_1} = \boldsymbol{tan^{-1}}\left(\frac{\boldsymbol{x_{TL}} - \boldsymbol{x_{frame}}}{\boldsymbol{y_{TL}} - \boldsymbol{y_{frame}}}\right) \times \frac{\boldsymbol{180}}{\boldsymbol{\pi}} \quad (2)$$

Distances in pixels are converted to distances in centimeters using linier regression.

$$y = A + Bx \quad (3)$$

Linear regression output, which shows the real distance in cm is y. *A is* coefficient A, *B is* coefficient B and x is input distance in pixel unit.

E. Estimation of Robot Position

Estimating the robot's position begins with finding the robot's orientation. Fig. 9 shows an illustration of calculating the robot's orientation with camera perception. An auxiliary line h_c is needed that connects point C to the center point of the robot. Point C is the midpoint of the line in front of the goal or the line connecting two L landmarks.

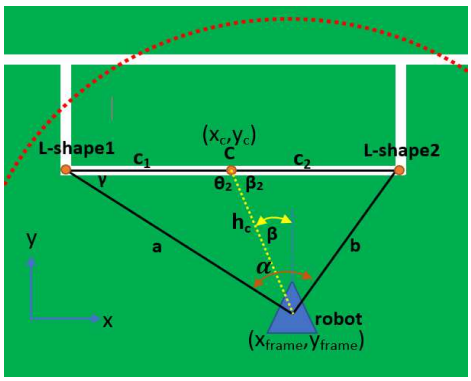

Fig. 9 Illustration to get robot orientation

Referring to the robot soccer field that has been determined, it is known that the length of line C is 300 cm. Line C is divided in half into c1 and c2. Both have the same length, which is 150 cm. The value of h_c can be obtained using the equation (5).

$$\alpha = \arccos\left(\frac{(b^2+a^2-c^2)}{2ba}\right) \quad (4)$$

$$h_c = \sqrt{c1^2 + a^2 - 2(c1 * a) * \cos(\gamma)} \quad (5)$$

$$\gamma = \arccos\left(\frac{(c^2+a^2-b^2)}{2Ca}\right) \quad (6)$$

Base on equation (4) - (6). line a is the distance between landmark L-shape 1 and robot. Line b connects landmark 2 with the center of the robot. Lines a and b produce an angle called α. θ_2 is formed from the angle of $\angle(L-shape1) \rightarrow C \rightarrow (Robot)$. θ_2 can be obtained through the equation (7). While β_2 angle is formed from angle of $Robot \rightarrow \angle C \rightarrow (L-shape2)$. The addition of θ_2 and β_2 angles produce a value of 180 as shown ini equation (8).

$$\theta_2 = \arccos\left(\frac{(c1^2+h_c{}^2-a^2)}{2(c1*h_c)}\right) \quad (7)$$

$$\beta_2 = 180 - \theta_2 \quad (8)$$

To estimate the robot's orientation level, the omnidirectional mirror image is divided into areas 1 to 4. The angle is mapped to be positive with a value of 0° to 180° and negative with a range between 0° to -180° as shown in Fig. 10. The positive corner is on the right and the negative corner is on the left.

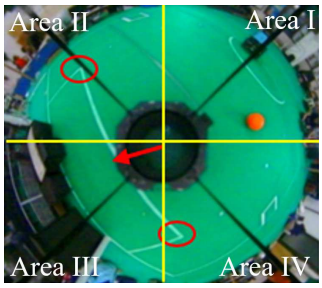

Fig. 10 Angle Mapping for Omnidirectional Images

β angle is formed from line h_c and robot orientation from camera perception or camera y axis. β value is provided via the equation (9). (x_c, y_c) is point C coordinate.

$$\beta = tan^{-1}\left(\frac{x_c - x_{frame}}{y_c - y_{frame}}\right) \quad (9)$$

$$\theta_{robot} = \beta \quad (10)$$

The orientation of the robot is made referring to point C. it's based on target kick on goal setting. So, the robot's orientation θ_{robot} is equal to **β**. If the robot is facing at point C, then it means the robot has an orientation of 0°. At this stage, the orientation of the robot has been brought to orientation to the field.

Furthermore, the position of the robot can be calculated based on the known landmark distances. Fig. 11 show an illustration for calculating the robot's position based on the landmark in front of the goal.

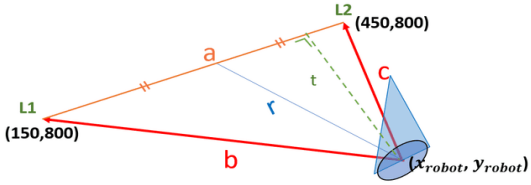

Fig. 11 Illustration of Robot Position Based on landmark

Whereas a is the penalty line's length in front of the goal, which is equal to 300 cm, b is distance of L1 landmark with robot center point (cm), c is distance of L2 landmark with robot center point (cm), t is the height of the right triangle between the robot and the L2 landmark with respect to the field (cm) and r is length of line between the robot's center and the penalty line's center (cm).

The robot's position must first be projected to calculate the robot's x-coordinate value and the robot's y-coordinate on the field. If Fig. 11 is projected on the x-axis and y-axis of the field, the results are shown in Fig. 12.

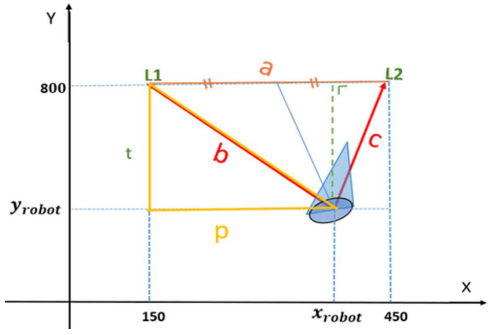

Fig. 12 The results of the projection of the robot's position on the field

$$s = \frac{a+b+c}{2} \quad (11)$$

$$\frac{1}{2}at = \sqrt{s\,(s-a)(s-b)(s-c)} \quad (12)$$

$$p = \sqrt{b^2 - t^2} \quad (13)$$

s is half the perimeter of the triangle. t is the height of the right triangle between the robot and the L2 landmark. So, to get the coordinates of the *y_robot* above the field:

$$y_{robot} = y_{L2} - t \quad (14)$$

$$x_{robot} = x_{L1} + p \quad (15)$$

y_{L2} is y-coordinate landmark L-shape L2 and x_{L1} is x-coordinate landmark L-shape L1. p is the length of the triangular base line between land mark L and the robot.

III. Result and Discussion

A. Model Performance Evaluation

The model was trained in Google Colab with Laptop Core i7 Gen 8 CPU @1.80GHz (8 core), 8GB RAM, OS Ubuntu 20.04 LTS 64 bit version, OpenCV library version 4.4.5. Inference run on ROS Noetic with same spec CPU. The distribution of training data and testing is 620 data for training data (include for validation) and 21 data for testing data. Data Validation, is used to validate the model and prevent overfitting. This strategy will combine training data with validation data. Validation dataset is data that has never been "seen" from the model. The training and validation process are carried out consecutively for each epoch or learning iteration. A validation procedure follows each training session.

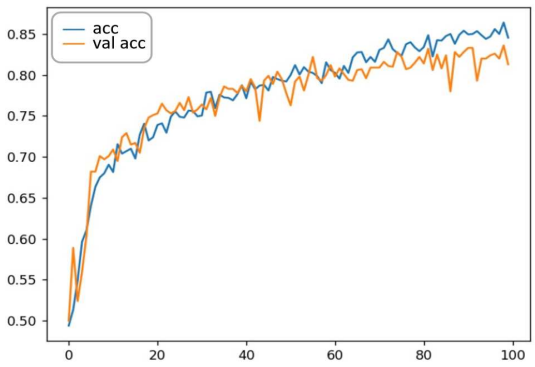

Fig. 13 Model and validation accuracy

Fig. 13 shows the results of the accuracy and validation accuracy of the resulting model. The horizontal axis represents the completed epoch. It can be seen that in the first epoch, the accuracy is less than 0.50 and so is the accuracy validation. The longer the training process, the fluctuations in accuracy and accuracy validation occur. However, the direction tends to increase closer to the value of 1. When the epoch is at the value of 60, the accuracy increases but the accuracy validation starts to lag behind. The train process is stopped at the 100th epoch to prevent overfitting. The accuracy value is 0.838 and the validation accuracy is 0.806. The results of the loss value and loss validation are shown in Fig. 14.

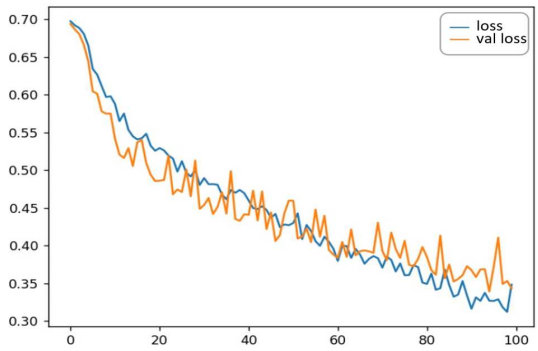

Fig. 14 Loss and validation loss from model

The presence of a loss value can make the model "learn" until the loss is reduced. The horizontal axis shows the running epoch. At the start of training, the loss value is 0.70 and the loss validation is 0.698. After the epoch increases, the loss value and validation loss tend to decrease. However, there are times when it increases, as in the 33rd epoch validation loss towards the 35th epoch, which increased in value from 0.461 to 0.516. The increase in the loss value also occurs at certain epochs. After the epoch reaches 100, the result is a loss value of 0.352 and a validation loss value of 0.35. Contrary to accuracy validation, if the loss value of the validation process increases, the training model is stopped to prevent overfitting.

B. Landmark L-shape Detection Result

This test is carried out to find out how far the landmark object can be detected. The results of the landmark detection test by the robot are shown in Table I. d_{LR} is distance of landmark L-shape to robot (cm). At a distance of 40-60 cm, the robot can detect both landmarks L-shape well. At a distance of 80 cm, the landmark L-shape 2 was recognized, but the landmark L-shape 1 was not. Up to a distance of 240 cm from the robot, the two landmarks L-shape are still easy to detect. Above that distance, L-shape landmarks are difficult to detect. According to visual observation, the white line looks small at distances above 240 cm. The further away, the smaller it looks. So, this reduces detection accuracy. In the field there are several robots that can communicate with each other. The robot closest to the landmark can share landmark detection information with other robots. Thus, detection results will be obtained that maintain the level of accuracy.

TABLE I
TESTING OF LANDMARK L-SHAPE DETECTION IN VARIOUS DISTANCES

d_{LR} (cm)	L-shape1	L-shape2
40	detected	detected
60	detected	detected
80	not detected	detected
100	detected	detected
120	detected	detected
140	detected	detected
160	detected	not detected
180	detected	detected
200	detected	detected
220	not detected	detected
240	detected	not detected
260	not detected	detected
280	not detected	not detected
300	not detected	not detected
320	not detected	not detected

The distance data in pixels have been converted into actual distance units using linear regression. The results of the conversion error are presented in Table II. d_{LRV} is distance of landmark L-shape to robot calculated by vision.

TABLE II
TESTING OF DISTANCE LANDMARK L-SHAPE TO ROBOT ESTIMATION

d_{LR} (cm)	d_{LRV} (cm)	*Error* (cm)
40	49	9
60	57	3
80	81	1
100	103	3
120	125	5
140	143	3
160	161	1
180	180	0
200	199	1
220	218	2
240	242	2
260	256	4
Error average		**2.83**

In this test, the image used is a sample image that successfully detects the L line landmark every 20 cm taken at a distance of 40 cm to 260 cm as much as 12 data. There is a difference in value between the actual distance and the distance measured by the robot using vision. The largest error value obtained is 9 cm, and the average error is 2.83 cm. In addition to the linear regression formula, which cannot produce 100% accuracy values, the results of the reflection of the hyperbolic omnidirectional mirror, and the added lens on the camera, most likely affect the accuracy of the conversion of pixel distances to actual distances.

C. Robot Position Data Estimation Result

Testing the position coordinates of the robot based on the localization of the sight using the L-shape reference of the landmark. The test location is in the penalty area field in front of the goal. The test results at this stage only use image data that has successfully detected two landmarks simultaneously. This is to find out how accurate the calculation results of the robot's coordinates are from the sight, because if the test includes image data that fails to detect two landmark lines, it will damage the data analysis.

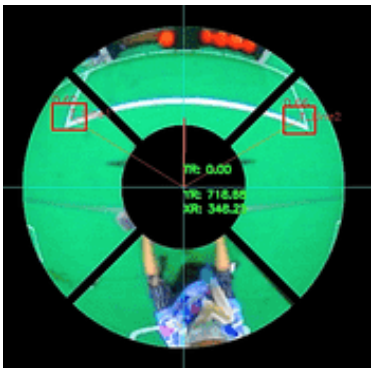

Fig. 15 Testing the robot's position at (350,716)

Fig. 15 presents a test of the robot's position at (350,716) cm. According to the estimation results made by the robot, the robot is at (348.25,718.68) cm. So, there is an error x of 2 cm and an error of y of 3 cm. Table III contains all of the test results in their entirety.

TABLE III
POSITION DATA ESTIMATION TESTING

Data No.	Position Estimation by vision (cm)		Real Position (cm)		*Error* (cm)	
	x_{robot}	y_{robot}	x_{real}	y_{real}	***Err x***	***Err y***
1	249.63	731.63	250	733	0.37	1.37
2	256.75	780.26	250	784	6.75	3.74
3	269.27	724.44	268	725	1.27	0.56
4	273.78	738.12	275	740	1.22	1.88
5	285.45	716.49	285	715	0.45	1.49
6	307.86	793.01	305	790	2.86	3.01
7	325.64	794.23	325	795	0.64	0.77
8	338.65	745.08	340	745	1.35	0.08
9	344.50	762.67	345	760	0.5	2.67
10	352.25	711.68	360	716	7.75	4.32
Error average (*Err x, Err y*):					2.32	1.99

The outcomes of evaluating the robot coordinates based on the computation of vision with reference to the detection of L-shape markers are shown in Table III. The test was carried out 10 times. The position of the x robot coordinates is set in a position range of 250 to 340 cm. The largest error value of the robot's x coordinate error generated by vision is 7.75 cm with an average error of 2.32 cm. The robot's Y coordinate test results are set in the 700 to 800 cm range. With an average error of 1.99 cm, the largest robot x coordinate error value produced by vision is 4.32 cm.

IV. CONCLUSION

From the results of the tests that have been carried out, there is an error in the estimation of the robot position data based on the landmark L-shape reference using vision. The biggest error in the estimation of the x position is 7.75 cm with an average error of 2.31 cm. While the biggest error in the estimation of the y position is 4.32 cm with an average error of 1.9 cm. The estimation of robot position data using rotary encoder-based odometry resulted in an average error of 22.07 cm for x and 20 cm for y [7]. Compared to this, the results of the estimation of robot position data based on landmark references using vision are considered more accurate. Conversion of pixels into real distances using linear regression still produces quite large errors. Where this conversion error will also affect the results on the estimation of the robot position data.

Testing and analyzing position estimation when the robot moved is recommended for further work. How is the accuracy of the robot position data estimation system based on landmark references using vision when robot moved. In addition, it is also necessary to try using other regression methods, such as Ab-Exponential regression or a fusion of linear regression and ab-exponential regression. To reduce pixel conversion error to real distance.

ACKNOWLEDGMENT

We thank PENS for funding and providing facilities for this research. And special thanks for ERSOW robotic team members that assisted in completing this study.

REFERENCES

[1] M. Jiono, Y. D. Mahandi, S. Norma Mustika, S. Sendari, and A. M. Dzikri, "Self Localization Based on Neighborhood Probability Mapping for Humanoid Robot," *4th Int. Conf. Vocat. Educ. Training, ICOVET 2020*, pp. 355–359, 2020, doi: 10.1109/ICOVET50258.2020.9230237.

[2] J. Palacín, E. Rubies, and E. Clotet, "Systematic Odometry Error Evaluation and Correction in a Human-Sized Three-Wheeled Omnidirectional Mobile Robot Using Flower-Shaped Calibration Trajectories," *Appl. Sci.*, 2022, doi: 10.3390/app12052606.

[3] F. Lui Hakim Ihsan, R. Adryantoro Priambudi, M. Mobed Bachtiar, and I. Kumianto Wibowo, "Heading Calibration in Robot Soccer ERSOW using Line Landmark on the Field," *IES 2020 - Int. Electron. Symp. Role Auton. Intell. Syst. Hum. Life Comf.*, pp. 226–232, 2020, doi: 10.1109/IES50839.2020.9231923.

[4] D. R. Phang, W. Lee, and P. Michail, "and IMU Sensor," *2019 IEEE Int. Meet. Futur. Electron Devices, Kansai*, pp. 2019–2020, 2019.

[5] M. A. Esfahani, H. Wang, K. Wu, and S. Yuan, "OriNet: Robust 3-D Orientation Estimation with a Single Particular IMU," *IEEE Robot. Autom. Lett.*, 2020, doi: 10.1109/LRA.2019.2959507.

[6] R. B. Sousa, M. R. Petry, and A. P. Moreira, "Evolution of odometry calibration methods for ground mobile robots," 2020, doi: 10.1109/ICARSC49921.2020.9096154.

[7] Z. T. Romadon, H. Oktavianto, I. K. Wibowo, B. Sena Bayu Dewantara, E. A. Nurrohmah, and R. Adryantoro Priambudi, "Pose Estimation on Soccer Robot using Data Fusion from Encoders, Inertial Sensor, and Image Data," in *IES 2019 - International Electronics Symposium: The Role of Techno-Intelligence in Creating an Open Energy System Towards Energy Democracy, Proceedings*, 2019, pp. 454–459, doi: 10.1109/ELECSYM.2019.8901578.

[8] M. B. Alatise and G. P. Hancke, "Pose estimation of a mobile robot based on fusion of IMU data and vision data using an extended kalman filter," *Sensors (Switzerland)*, vol. 17, no. 10, 2017, doi: 10.3390/s17102164.

[9] S. Luo, H. Lu, J. Xiao, Q. Yu, and Z. Zheng, "Robot detection and localization based on deep learning," *Proc. - 2017 Chinese Autom. Congr. CAC 2017*, vol. 2017-Janua, pp. 7091–7095, 2017, doi: 10.1109/CAC.2017.8244056.

[10] R. Alves, J. Silva De Morais, and K. Yamanaka, "Cost-effective indoor localization for autonomous robots using kinect and wifi sensors," *Intel. Artif.*, 2020, doi: 10.4114/intartif.vol23iss65pp33-55.

[11] M. Karkoub, O. Bouhali, and A. Sheharyar, "Gas Pipeline Inspection Using Autonomous Robots with Omni-Directional Cameras," *IEEE Sens. J.*, vol. 21, no. 14, pp. 15544–15553, 2021, doi: 10.1109/JSEN.2020.3043277.

[12] S. Barone, M. Carulli, P. Neri, A. Paoli, and A. V. Razionale, "An omnidirectional vision sensor based on a spherical mirror catadioptric system," *Sensors (Switzerland)*, 2018, doi: 10.3390/s18020408.

[13] N. Engel, S. Hoermann, M. Horn, V. Belagiannis, and K. Dietmayer, "DeepLocalization : Landmark-based Self-Localization with Deep Neural Networks," *2019 IEEE Intell. Transp. Syst. Conf.*, pp. 926–933, 2019.

[14] B. N. Krishna Sai and T. Sasikala, "Object Detection and Count of Objects in Image using Tensor Flow Object Detection API," *Proc. 2nd Int. Conf. Smart Syst. Inven. Technol. ICSSIT 2019*, no. Icssit, pp. 542–546, 2019, doi: 10.1109/ICSSIT46314.2019.8987942.

[15] D. Xianzhi, "Research on Camera Calibration Technology Based on Deep Neural Network in Mine Environment," *Proc. - 2020 Int. Conf. Comput. Vision, Image Deep Learn. CVIDL 2020*, no. Cvidl, pp. 375–379, 2020, doi: 10.1109/CVIDL51233.2020.00-68.

[16] M. R. Dwijayanto, S. Kurniawan, and B. Sugandi, "Real-Time Object Recognition for Football Field Landmark Detection Based on Deep Neural Networks," *Proc. 2019 2nd Int. Conf. Appl. Eng. ICAE 2019*, 2019, doi: 10.1109/ICAE47758.2019.9221678.

[17] A. F. Villán, "Mastering OpenCV 4 with Python: A Practical Guide Covering Topics from Image Processing, Augmented Reality to Deep Learning with OpenCV 4 and Python 3.7," *Packt Publishing*. 2019.

[18] M. J. J. Douglass, "Book Review: Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow, 2nd edition by Aurélien Géron," *Phys. Eng. Sci. Med.*, 2020, doi: 10.1007/s13246-020-00913-z.

[19] A. M. Taqi, F. Al-Azzo, A. Awad, and M. Milanova, "Skin Lesion Detection by Android Camera based on SSD-Mo-bilenet and TensorFlow Object Detection API," *Am. J. Adv. Res.*, 2019, doi: 10.5281/zenodo.3264022.

[20] T. V. Janahiraman and M. S. M. Subuhan, "Traffic light detection using tensorflow object detection framework," 2019, doi: 10.1109/ICSEngT.2019.8906486.

[21] M. Abagiu, D. Popescu, F. L. Manta, and L. C. Popescu, "Use of a Deep Neural Network for Object Detection in a Mobile Robot Application," *EPE 2020 - Proc. 2020 11th Int. Conf. Expo. Electr. Power Eng.*, no. Epe, pp. 221–225, 2020, doi: 10.1109/EPE50722.2020.9305648.

[22] L. Jing and Y. Tian, "Self-Supervised Visual Feature Learning with Deep Neural Networks: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2021, doi: 10.1109/TPAMI.2020.2992393.

[23] Y. Koo and S. H. Kim, "Distributed Logging System for ROS-based Systems," *2019 IEEE Int. Conf. Big Data Smart Comput. BigComp 2019 - Proc.*, pp. 1–3, 2019, doi: 10.1109/BIGCOMP.2019.8679372.

[24] M. Marian, F. Stinga, M. T. Georgescu, H. Roibu, D. Popescu, and F. Manta, "A ROS-based Control Application for a Robotic Platform Using the Gazebo 3D Simulator," *Proc. 2020 21st Int. Carpathian Control Conf. ICCC 2020*, 2020, doi: 10.1109/ICCC49264.2020.9257256.

[25] A. Y. R. Ruiz and B. Chandrasekaran, "A Robotic Control System Using Robot Operating System and MATLAB for Sensor Fusion and Human-Robot Interaction," *2020 10th Annu. Comput. Commun. Work. Conf. CCWC 2020*, pp. 550–555, 2020, doi: 10.1109/CCWC47524.2020.9031184.

[26] T. Witte and M. Tichy, "Checking consistency of robot software architectures in ROS," *Proc. - Int. Conf. Softw. Eng.*, pp. 1–8, 2018, doi: 10.1145/3196558.3196559.

[27] X. Liu *et al.*, "Self-supervised Learning: Generative or Contrastive," *IEEE Trans. Knowl. Data Eng.*, 2021, doi: 10.1109/TKDE.2021.3090866.

[28] F. Zhang, Q. Li, Y. Ren, H. Xu, Y. Song, and S. Liu, "An expression recognition method on robots based on mobilenet V2-SSD," *2019 6th Int. Conf. Syst. Informatics, ICSAI 2019*, no. Icsai, pp. 118–122, 2019, doi: 10.1109/ICSAI48974.2019.9010173.

[29] G. Yu, L. Wang, M. Hou, Y. Liang, and T. He, "An adaptive dead fish detection approach using SSD-MobileNet," *Proc. - 2020 Chinese Autom. Congr. CAC 2020*, no. 2018, pp. 1973–1979, 2020, doi: 10.1109/CAC51589.2020.9326648.

[30] Y. Qian, R. Jiacheng, W. Pengbo, Y. Zhan, and G. Changxing, "Real-Time detection and localization using SSD method for oyster mushroom picking robot∗," *2020 IEEE Int. Conf. Real-Time Comput. Robot. RCAR 2020*, pp. 158–163, 2020, doi: 10.1109/RCAR49640.2020.9303258.

[31] X. Song, P. Jiang, and H. Zhu, "Research on Unmanned Vessel Surface Object Detection Based on Fusion of SSD and Faster-RCNN," *Proc. - 2019 Chinese Autom. Congr. CAC 2019*, pp. 3784–3788, 2019, doi: 10.1109/CAC48633.2019.8997431.